AMENDMENTS TO THE CLAIMS

1. (Currently amended) A method implemented by a computer system for managing a run queue comprising a first plurality of threads having a sortable priority that are sorted with respect to one another based on thread priority, the method comprising:

in a deterministic amount of time equivalent to an amount of time to insert a single thread into the run queue, associating a second plurality of threads from a sleep queue that is priority sorted within the run queue in a manner that maintains a sortable priority of the sleep queue based scheduling semantic of the run queue wherein associating the second plurality of threads with the run queue further includes inserting only a root thread of the second plurality of threads into the run queue; and

executing respective ones of the threads in view of thread priority.

2. (Previously presented) The method of claim 1, wherein the second plurality of threads comprises a root thread, and wherein associating the second plurality of threads with the run queue further comprises inserting only the root thread into the run queue.

3. (Previously presented) The method of claim 1, wherein the associating the second plurality of threads with the run queue further comprises inserting each thread in the second plurality of threads into the run queue independent of any additional other queue access.

4. (Cancelled).

5. (currently amended) The method of claim 1, ~~wherein associating the second plurality of threads with the run queue further comprises:~~

~~inserting only a root thread of the second plurality of threads into the run queue; and~~

wherein the method further comprises:

removing the root thread from the run queue; and

responsive to removing the root thread, inserting a next thread of the second plurality of threads into the run queue such that the priority based scheduling semantic of the run queue is preserved.

6. (Previously presented) The method of claim 1, wherein the method further comprises:

inserting a root thread of the second plurality of threads into the run queue;

removing the root thread from the run queue for execution; and

responsive to removing the root thread and independent of any additional other queue access, inserting a next thread of the second plurality of threads into the run queue.

7. (Canceled).

8. (Currently amended) A system for managing a run queue, the run queue comprising a first plurality of threads, each thread in the first plurality of

threads having a respective priority such that the threads is sortable, one to another, the first plurality of threads being sorted such that a thread having a high priority is removed from the run queue before a thread having a lower priority, the system comprising:

memory for storing the run queue and computer-executable instructions;

a processor operatively coupled to the memory, the processor being configured to execute the computer-executable instructions for:

in an amount of time to insert a single thread into the run queue, associating a ~~the~~ second plurality of threads from a sleep queue that is priority sorted within the run queue, the association maintains ~~associating maintaining~~ a priority based scheduling semantic of the run queue, wherein associating the second plurality of threads with the run queue further includes inserting only a root thread of the second plurality of threads into the run queue;

removing the root thread from the run queue; and

responsive to removing the root thread, inserting a next thread of the second plurality of threads into the run queue such that the priority based scheduling semantic of the run queue is preserved; and

executing respective ones of the threads in view of thread priority.

9. (Previously presented) The system of claim 8, wherein associating the second plurality of threads with the run queue is performed independent of more than a single other queue access.

10. (Currently amended) The system of claim 8, wherein the second plurality of threads comprises ~~a~~ the root thread operatively coupled to one or more other threads of the second plurality of threads, each of the one or more other threads having a respective priority that is a lower priority or an equal priority as compared to a priority of the root thread.

11. (Cancelled).

12. (Cancelled).

13. (previously presented) The system of claim 8:

wherein the first plurality of threads is a first linked list data structure;

wherein the second plurality of threads is a second linked list data structure comprising the root node that is operatively coupled to one or more other threads in the second plurality of threads; and

wherein the single insert operation is an operation comprising inserting the root node into a position in the first linked list data structure.

14. (Cancelled).

15. (Cancelled).

16. (Currently amended) A computer storage media comprising computer-program instructions executable by a processor to manage a run queue of executable threads that are sortable with respect to one another based on thread priority, the computer-program instructions when executed by the processor implementing operations comprising:

in a deterministic amount of time that is independent of the number of threads in a second plurality of threads from a sleep queue that is priority sorted, the deterministic amount of time being a time to insert a single thread into the run queue, associating the second plurality of threads with a first plurality of threads in the run queue in a manner that maintains a priority based scheduling semantic of the run queue, wherein associating the second plurality of threads with the run queue further includes inserting only a root thread of the second plurality of threads into the run queue;

removing the root thread from the run queue;

responsive to removing the root thread, inserting a next thread of the second plurality of threads into the run queue such that the priority based scheduling semantic of the run queue is preserved; and

executing respective ones of the threads in view of thread priority.

17. (Previously presented) The computer storage media of claim 16, wherein the second plurality of threads comprises a root thread that is operatively coupled to one or more other threads of the second plurality of threads, and wherein the computer-program instructions for associating further comprise instructions for inserting only the root thread into the first plurality of threads.

18. (Previously presented) The computer storage media of claim 16, wherein the first plurality of threads is a first linked list data structure, the second plurality of threads is a second linked list data structure comprising a root node that is operatively coupled to one or more other threads in the second plurality of threads, and the deterministic amount of time is a result of a single insert operation to insert the root node into the first linked list data structure.

19. (Cancelled).

20. (Currently amended) The computer storage media of claim 16 ~~19~~, wherein the computer-program instructions for inserting the next thread are performed independent of an other queue.

21. (Cancelled).

22. (Cancelled).

23. (Currently amended) A computer storage media comprising computer-program instructions executable by a processor for:

managing a run queue with a run queue data structure, the run queue data structure comprising:

a first dimension data field comprising a first plurality of threads that are sortable, one to another, with respect to thread priority; and

a second dimension data field comprising a second plurality of threads from a sleep queue that are sortable, one to another, based on thread priority of the sleep queue, the second plurality of threads comprising a root thread and one or more other threads, wherein the second plurality of threads are associable with the run queue by inserting only a root thread of the second plurality of threads into the run queue; and

executing respective ones of the threads in view of the thread priority.

24. (Cancelled).